COMP9414 考点复习

`UNSW` `COMP9414`

Outline 哦
- **Course_name** ：COMP9414
- **Lecturer** ：Alan Blair
- **Author** ： Zeshi Wu

---

1. Environment/Agent Types

- 5 types of agent:
  - **Reactive Agent** :do not have memory
  - **Model-Based Agent** : have past memory, but cannot planning base on the past
  - **Planning Agent** : can plan the furture，but not flexible in adapting to new situations.
  - **Game Playing Agent**: have opponent

- Learning Agent:

2. Prolog Programming

- 编程 递归 sort 看实验

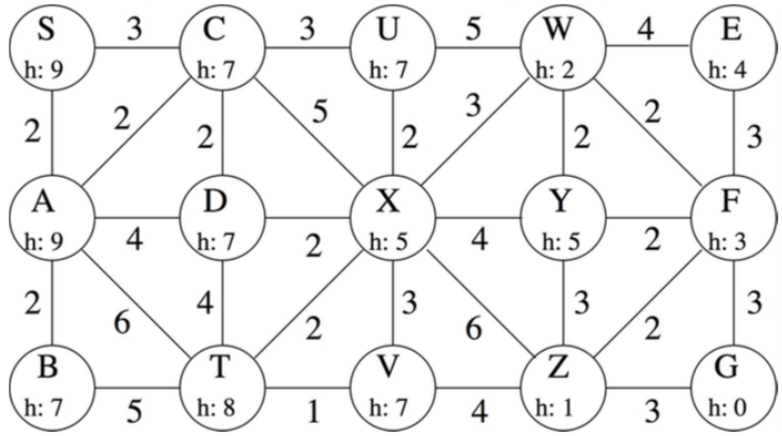
3. Path Search

- 用state 来 Formulate the problem precisely
- toturial
  - Generally, BFS is a bit faster than DFS 随机地图
  - 同心地图，DFS 更快
  - 找一个DFS 比 BFS更快的图
    - This would be true for a search problem where there exist many solutions at a fairly deep level, such that DFS will find a solution more or less as soon as it gets to that depth for the first time. For example, imagine you are shipwrecked in the middle of a lake and you need to find a path to the shoreline, with a series of small steps either up, down, left or right. DFS will find a path quickly, whereas BFS will spend a lot of time exploring all paths shorter than the radius of the lake.

4. Heuristic Path Search 看一下hw3

- 概念
  - 曼哈顿距离
  - XX距离
- toturial

Consider the task of finding a path from start state S to goal state G, given the distances and heuristic values in this diagram:



For each of the following strategies, list the order in which the states are expanded. Whenever there is a choice of states, you should select the one that comes first in alphabetical order. In each case, you should skip any states that have previously been expanded, and you should continue the search until the goal node is expanded.

a. Breadth First Search

S, A, C, B, D, T, U, X, V, W, Y, Z, E, F, G

b. Depth First Search

S, A, B, T, D, C, U, W, E, F, G

c. Uniform Cost Search [Hint: first compute g() for each state in the graph]

S(0), A(2), C(3), B(4), D(5), U(6), X(7), T(8), V(9), W(10), Y(11), F(12), Z(13), E(14), G(15)

d. Greedy Search, using the heuristic shown

S, C, X, Z, G

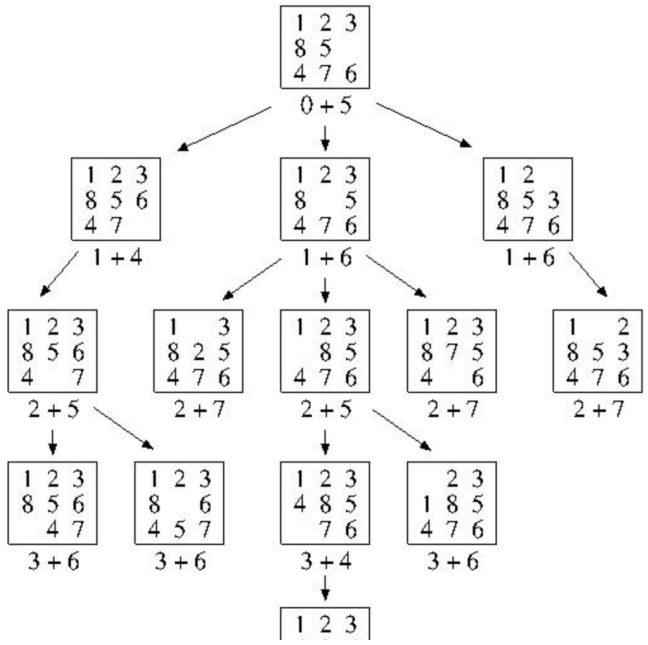
e. A*Search, using the heuristic shown

S(9), C(10), A(11), B(11), D(12), X(12), W(12), U(13), Z(14), F(15), G(15)

Note that g(X) becomes 8 after C is expanded, but drops to 7 after D is expanded. Similarly, g(G) becomes 16 when Z is expanded, but drops to 15 after F is expanded.

- UCS 用dj 算法。也算是贪婪 求出所有最小的G（ ）
- GREED 每次走最小的H（ ）
- A*， UCS求完所有最小的G（ ），然后加上H（ ），就是最后的数

Trace the A*Search algorithm using the Total Manhattan Distance heuristic, to find the shortest path from the initial state shown above, to the goal state.



  - G ( ) + H ( )

- 总结
  - Heuristics 可以降低搜索成本。
  - Greedy 试图每次走最小的H ( )
  - A^* 结合UCS & greedy
  - A^* 好棒棒：）complete, optimal and optimally efficient among all optimal search algorithms.
  - 内存的使用仍然是问题
    Memory usage is still a concern for A^*. IDA^* is a low-memory variant.
  - BFS 是UCS的特殊情况
    - Uniform Cost Search reduces to Breadth First Search when all edges have the same cost.
  - BFS，DFS，UCS是best-first search的特殊情况
    - best-first search reduces to BFS when f(n)=number of edges from start node to n,
    - to UCS when f(n)=g(n);
    - it can be reduced to DFS by, for example, setting f(n)=-(number of nodes from start state to n) (thus forcing deep nodes on the current branch to be searched before shallow nodes on other branches). Another way to produce DFS is to set f(n)=-g(n) (but this might produce a particularly bad choice of nodes within the DFS framework

        - for example, try tracing the order in which nodes are expanded when traveling from Urziceni to Craiova).
    - Uniform Cost Search is a special case of A*Search
        - A*Search reduces to UCS when the heuristic function is zero everywhere, i.e. h(n)=0 for all n;
          this heuristic is clearly admissible since it always (grossly!) underestimates the distance remaining to reach the goal.

The **heuristic path algorithm** is a best-first search in which the objective function is

$$f(n) = (2-w)g(n) + wh(n)$$

What kind of search does this perform when w=0? when w=1? when w=2?

This algorithm reduces to Uniform Cost Search when w=0, to A* Search when w=1 and to Greedy Search when w=2.

For what values of w is this algorithm complete? For what values of w is it optimal, assuming $h()$ is admissible?

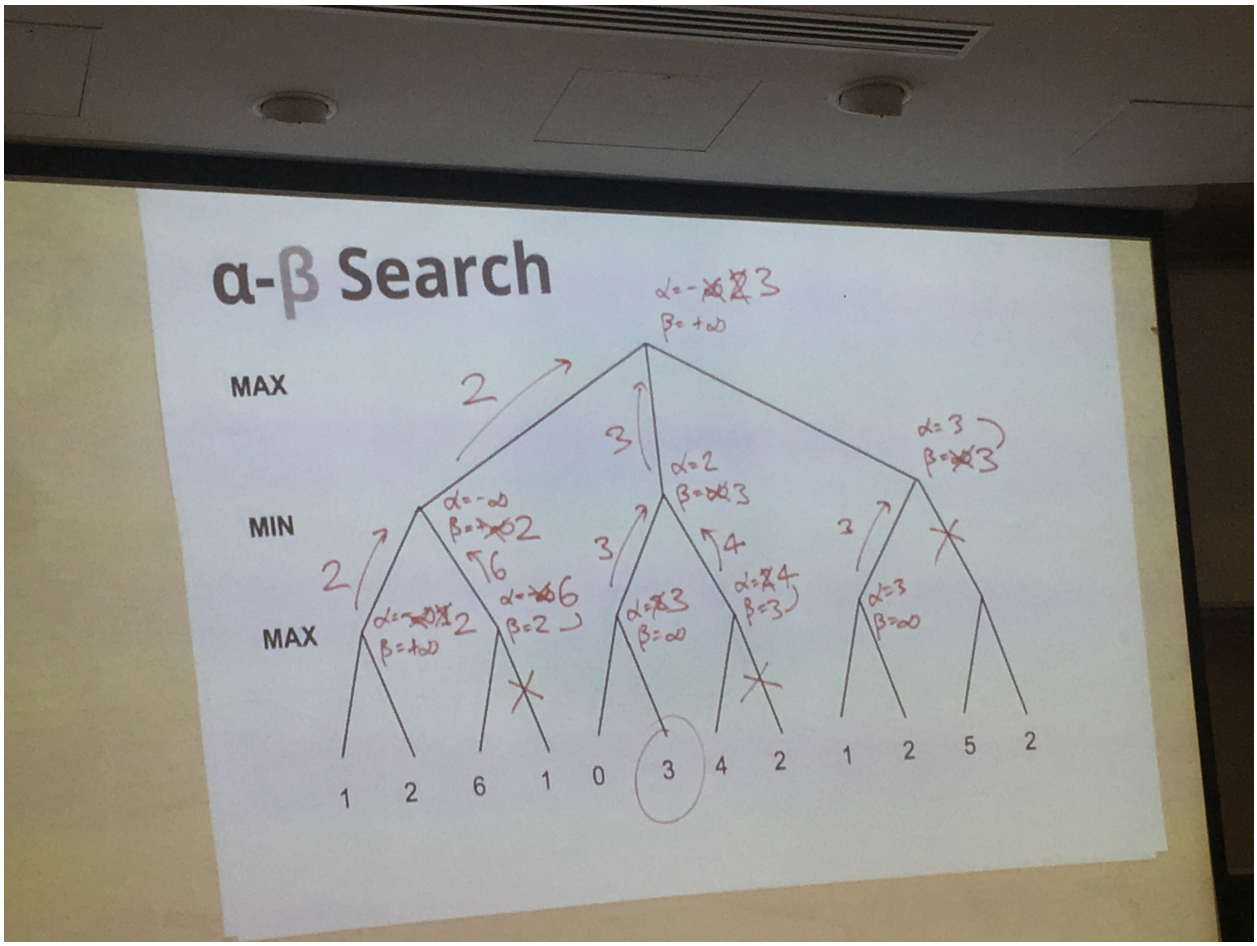
It is guaranteed to be optimal when $0 \le w \le 1$, because it is equivalent to A* Search using the heuristic

$$h'(n) = [w/(2-w)]h(n) \le h(n)$$

When $w > 1$ it is not guaranteed to be optimal (however, it might work very well in practice, for some problems).

5. Game Playing

- $\alpha - \beta$

- 

- 很可能考 Wumpus 那个傻逼游戏

6. Constraint Satisfaction

- 不同点

Path Search vs. Constraint Satisfaction

Important difference between Path Search Problems and CSP's:

- Constraint Satisfaction Problems (e.g. n-Queens)
  - difficult part is knowing the final state
  - how to get there is easy
- Path Search Problems (e.g. Rubik's Cube)
  - knowing the final state is easy
  - difficult part is how to get there

- backtracking search and heuristics
  - 几个做法
    - Minimum Remaining Values 选fewest legal values.
    - Degree Heuristic 选most constraints
- forward checking and arc consistency 正向检测与弧一致性
  - forward checking （Constraint propagation约束传播）
    - **Activity 6.2** Forward Checking and Arc Consistency

      Use Forward Checking to show that the Australia map-colouring problem has no solution when we assign WA=green, V=Red, NT=Red. If the tree?

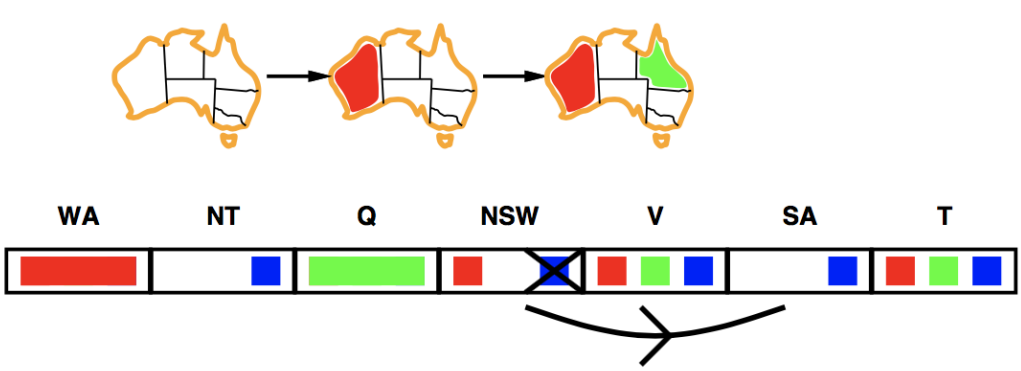
	WA	NT	Q	NSW	V	SA	T
initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
WA=Green	G	R B	R G B	R G B	R G B	R B	R G B
V = Red	G	R B	R G B	G B	R	B	R G B
NT = Red	G	R	G B	G B	R	B	R G B
SA = Blue	G	R	G	G	R	B	R G B
Q = Green	G	R	G		R	B	R G B

      No options remain for NSW, so there is no solution.
  - arc consistency

Arc consistency

Simplest form of propagation makes each arc consistent

$X \rightarrow Y$ is consistent if
for every value x of X there is some allowed y

- 

- 考题
  - 加减问题
  - 上色

7. Logical Agents

P	Q	¬ P	P ∧ Q	P ∨ Q	P ⇒ Q
F	F	T	F	F	T
F	T	T	F	T	T
T	F	F	F	T	F
T	T	F	T	T	T

Syntax of First Order Logic

Constants	$Gold, Wumpus, [1,2], [3,1]$, etc.
Predicates	$Adjacent(), Smell(), Breeze(), At()$
Functions	$Result()$
Variables	$x, y, a, t, \ldots$
Connectives	$\wedge \vee \neg \Rightarrow \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall \exists$

- 考题： 写表达式 / 翻译

  c. (Smoke ⇒ Fire) ⇒ (¬ Smoke ⇒ ¬ Fire)

  Satisfiable

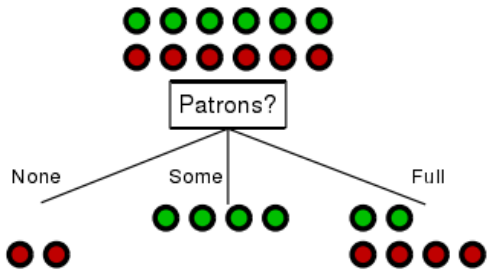
Smoke	Fire	Smoke ⇒ Fire	¬ Smoke ⇒ ¬ Fire	KB
T	T	T	T	T
T	F	F	T	T
F	T	T	F	F
F	F	T	T	T

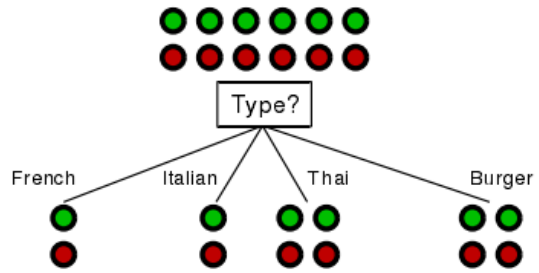
    Models are: {Smoke, Fire}, {Fire}, {}

    - KB是什么？

8. Learning and Decision Trees

- 求熵 Entropy
  - 因为训练数据有noise，使之不一致，为了确保产生正确的树，引入熵





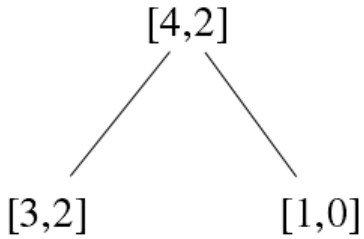
$$\begin{aligned}\text{For Patrons, Entropy} &= \frac{1}{6}(0)+\frac{1}{3}(0)+\frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3})-\frac{2}{3}\log(\frac{2}{3})\right] \\ &= 0+0+\frac{1}{2}\left[\frac{1}{3}(1.585)+\frac{2}{3}(0.585)\right]=0.459 \\ \text{For Type, Entropy} &= \frac{1}{6}(1)+\frac{1}{6}(1)+\frac{1}{3}(1)+\frac{1}{3}(1)=1\end{aligned}$$

  - 图中Patrons attring比type要好，因为其分类的三个set中有两个是完全集
  - 熵越大代表信息越混乱，所以E越小越好
- Induce Tree (Pruning剪枝 Laplace Error)
  - 根据奥卡姆剃刀, 我们需要对那些对分类增益不大的分支进行修剪
  - $E=1-\frac{n+1}{N+K}$
  - N = 总结点数
  - n = number of items in the majority class
  - K = number of classes
  - 如果子节点的平均拉普拉斯误差超过父节点->剪枝
  - e.g.1

Should the children of this node be pruned or not?

Left child has class frequencies [3,2]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{3+1}{5+2} = 0.429$$



- Right child has $E = 0.333$

  Parent node has $E = 0.375$

  Average for Left and Right child is

$$E = \frac{5}{6}(0.429) + \frac{1}{6}(0.333) = 0.413$$

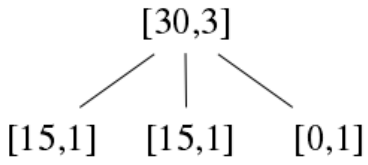
Since $0.413 > 0.375$, children should be pruned.

- e.g.2

Should the children of this node be pruned or not?

Left and Middle child have class frequencies [15,1]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{15+1}{16+2} = 0.111$$



- Right child has $E = 0.333$

  Parent node has $E = \frac{4}{35} = 0.114$

  Average for Left, Middle and Right child is

$$E = \frac{16}{33}(0.111) + \frac{16}{33}(0.111) + \frac{1}{33}(0.333) = 0.118$$

Since $0.118 > 0.114$, children should be pruned.

9. Perceptrons and Neural Networks

感知器和神经网络讲的都比较基础，拓展可以看COMP9417课件

- NN模型
  - input edges

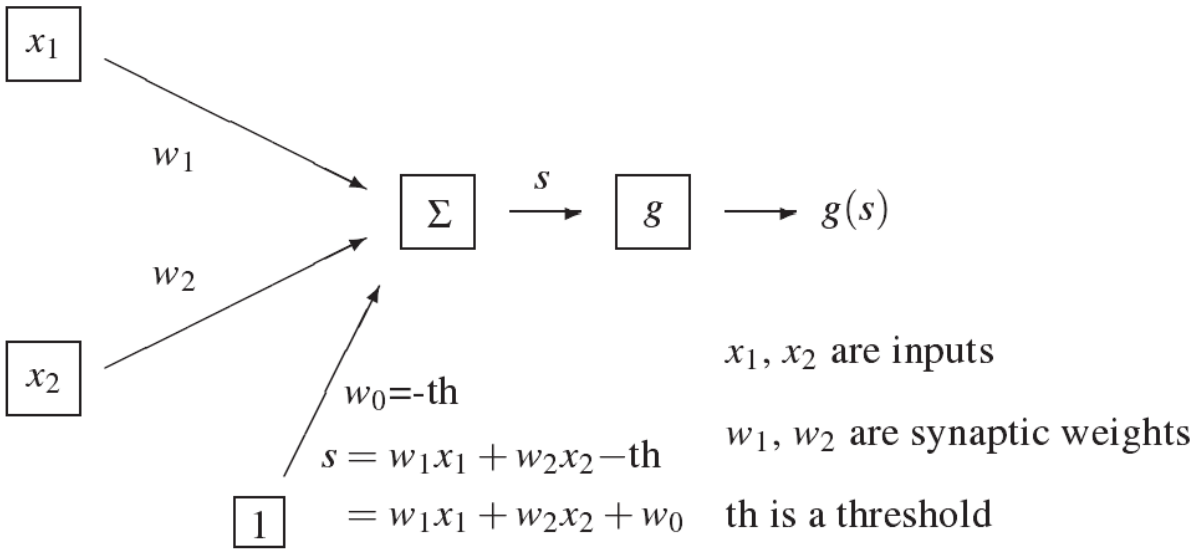
- output edges
- an activaion level(输入_激活函数)

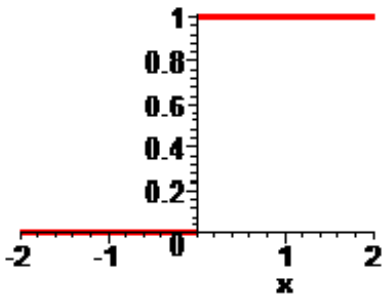
  权重可以是+的或-的，并可能随着时间的推移改变（学习）。

  输入函数是activation level of inputs 的加权和。

  激活层是一个**非线性传递**函数

  $$activaiton_i = g(s_i) = g(\sum_j w_{ij}x_j)$$

- 感知器
  - 

    - 采用的感知器为二元线性分类器，图为单层人工神经网络
    - g 是转换函数（通常为sigmoid函数），这门课用二元分类函数
    - 

      $$g(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ 0, & \text{if } s < 0 \end{cases}$$

    - Linear Separability 线性可分性

- AND $w_1 = w_2 = 1.0, \quad w_0 = -1.5$
- OR $w_1 = w_2 = 1.0, \quad w_0 = -0.5$
- NOR $w_1 = w_2 = -1.0, \quad w_0 = 0.5$

- 学习规则，改变权重

  Adjust the weights as each input is presented.

  recall: $s = w_1 x_1 + w_2 x_2 + w_0$

  if $g(s) = 0$ but should be 1,

  $$\begin{aligned} w_k &\leftarrow w_k + \eta\, x_k \\ w_0 &\leftarrow w_0 + \eta \\ \text{so} \quad s &\leftarrow s + \eta\,(1 + \sum_k x_k^2) \end{aligned}$$

  if $g(s) = 1$ but should be 0,

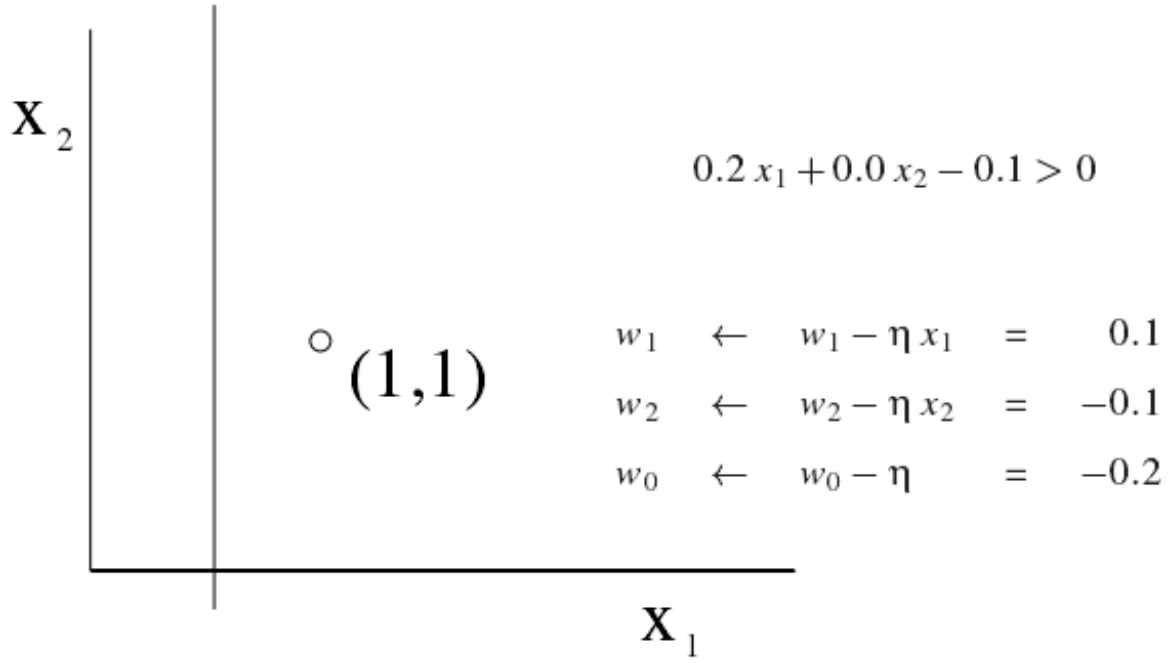
  $$\begin{aligned} w_k &\leftarrow w_k - \eta\, x_k \\ w_0 &\leftarrow w_0 - \eta \\ \text{so} \quad s &\leftarrow s - \eta\,(1 + \sum_k x_k^2) \end{aligned}$$

  otherwise, weights are unchanged. ($\eta > 0$ is called the **learning rate**)

  Theorem: This will eventually learn to classify the data correctly, as long as they are **linearly separable**.

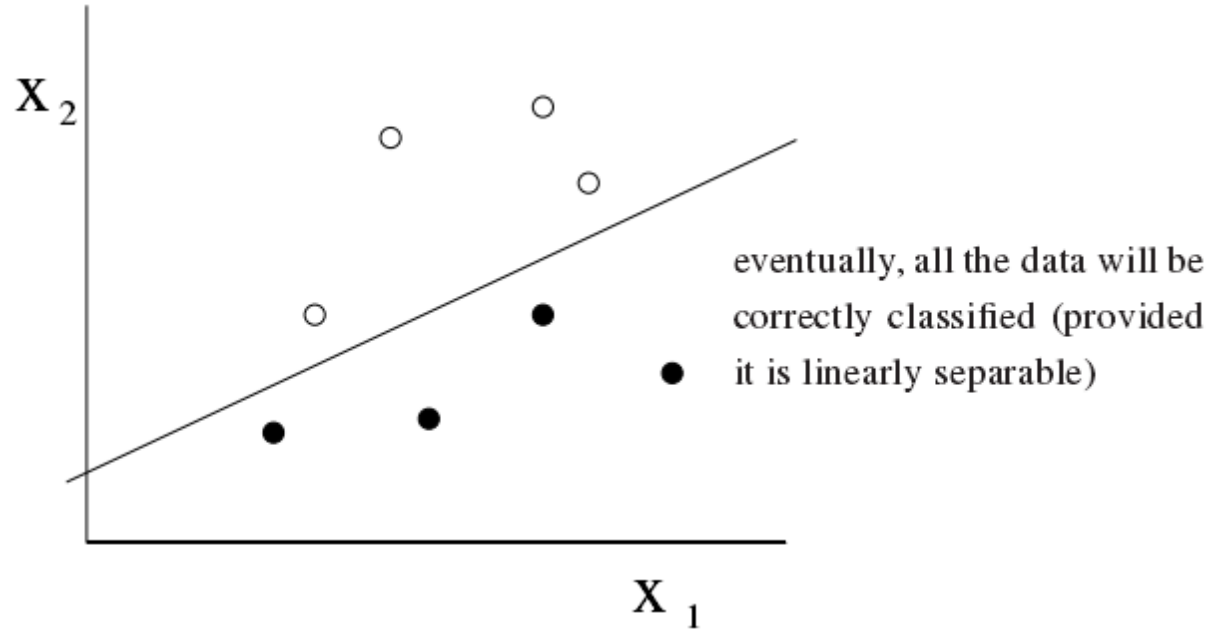
  - η *通常为0.1或1*

- 举个例子
  - 判断函数 $w_1 x_1 + w_2 x_2 + w_0 > 0$
  - 学习速率 η =0.1
  - 初始化随机权重 $w_1 = 0.2, w_2 = 0.1, w_0 = -0.1$

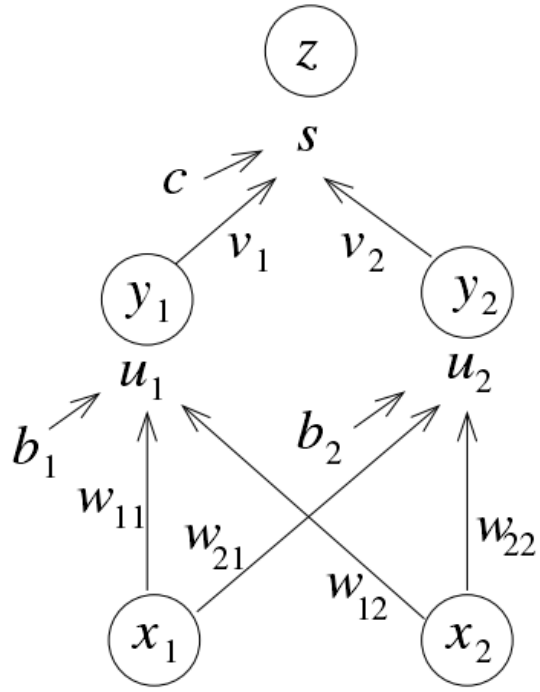


图中(1,1)不为实心，代表为负(判断函数为负)，但是将(1,1)代入函数为正，所以修改权重

- 最终所有结果被正确分类（*最后的权重为图中的分界线*）



- Multi—Layer Nerual Netwrok
  - 解决XOR异或问题
  - 首先将感知器中的二元分类函数变为连续函数，如sigmoid函数
  - variance方差，cost function(error function)= $\frac{1}{2}\sum(y_i - \hat{y_i})^2$
  - 梯度下降，获得最小的E

- 

$$
\begin{aligned}
u_1 &= b_1 + w_{11}x_1 + w_{12}x_2 \\
y_1 &= g(u_1) \\
s &= c + v_1 y_1 + v_2 y_2 \\
z &= g(s) \\
E &= \frac{1}{2}\sum (z-t)^2
\end{aligned}
$$

- 总结
  - 输入数据和输出结果都需要归一化
  - 权重随机初始化very small 的值（eta为0.1，权重初始值太多会很耗时）
  - online learning / batch learning
  - 防止过拟合
    - 简化模型
    - 交叉检验 cross validation
    - weight decay
  - 在梯度下载的过程中可调整η值

10. Reasoning under Uncertainty

- 概念

概率	描述
先 验 概 率	事件发生前的预判概率。可以是基于历史数据的统计，可以由背景常识得出，也可以 是人的主观观点给出。一般都是单独事件概率，如P(x),P(y)
后 验 概 率	事件发生后求的反向条件概率, 或者说，基于先验概率求得的反向条件概率。概率形式与条件概率相同

概率	描述
条件概率	一个事件发生后另一个事件发生的概率。一般的形式为$P(x\|y)$, 表示y发生前提下x发生 的概率
似然概率	通过历史数据统计得到的条件概率, 一般不把它叫做先验概率，但从定义上也符合先验定义

- $$P(Cause|Effect) = \frac{P(Effect|Cause)P(Cause)}{P(Effect)}$$
  - P(Cause|Effect)为后验概率, 即为求解目标
  - P(Effect|Cause)为似然概率
  - P(Cause)为先验概率
  - P(Effect) 其实也是先验概率，只是在贝叶斯的很多应用中不重要(因为只要最大后验不求绝 对值)，需要时往往用全概率公式计算得到
  - 最大似然理论: 令似然函数P(x|y)最大的y'即为y的最大似然估计

$$Max(P(x|y)) = \prod_{k=1}^{n} P(x_k|y) \text{ , for all y}$$

- - 贝叶斯理论: 在利用总体信息和样本信息的同时, 还利用先验概率p(y)
    - 因为有可能某个y是很稀有的类别几千年才看见一次，即使P(x|y)很高，也很可能不是它
    - 贝叶斯理论存在的问题: 实践中先验概率可能并不准确

$$y = Max(P(x|y)P(y))$$

- Inference

Start with the joint distribution:

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	**.108**	**.012**	**.072**	**.008**
¬ *cavity*	**.016**	**.064**	**.144**	**.576**

For any proposition ϕ, sum the atomic events where it is true:

$P(\phi) = \sum_{\omega:\omega \models \phi} P(\omega)$

$P(toothache) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$

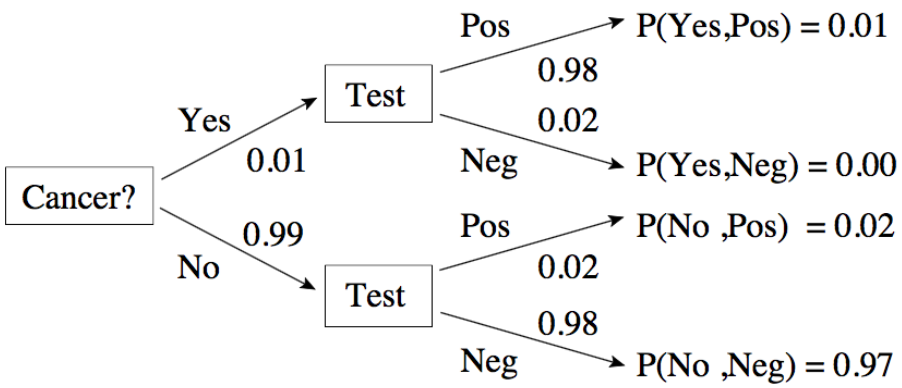
-

	toothache		¬ *toothache*	
	catch	¬ *catch*	*catch*	¬ *catch*
cavity	**.108**	**.012**	**.072**	**.008**
¬ *cavity*	**.016**	**.064**	**.144**	**.576**

Can also compute conditional probabilities:

$$P(\neg cavity | toothache) = \frac{P(\neg \texttt{cavity} \wedge \texttt{toothache})}{P(\texttt{toothache})}$$
$$= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4$$

- Bayes’ Rule and Conditional Independence



$$P(\texttt{cancer}\,|\,\texttt{positive}) = \frac{P(\texttt{positive}\,|\,\texttt{cancer})P(\texttt{cancer})}{P(\texttt{positive})}$$

$$= \frac{0.98 * 0.01}{0.98 * 0.01 + 0.2 * 0.99} = \frac{0.01}{0.01 + 0.02} = \frac{1}{3}$$